

# Comprehensive SystemC

## Standard Level - 5 days

Comprehensive SystemC is a 5-day training course introducing SystemC™, a C++ class library for system-level modeling. SystemC is typically used to model systems that have both hardware and software content at the transaction level of abstraction.

The syllabus covers the SystemC core language and its application to transaction-level modeling. The course complies with IEEE 1666-2005 and the SystemC 2.2 class library

**Comprising 2 modules**, engineers can attend either the full 5-day course or just the Fundamentals of SystemC module. Attendance of both modules is recommended unless attendees already have a good background in the use of C++.

- **Essential C++ for SystemC (days 1-2)** takes engineers who have a basic knowledge of the C programming language and gives them a fast-track way to acquire a good grounding in C++, which is an essential foundation for learning SystemC.
- **Fundamentals of SystemC (days 3-5)** builds on the foundation laid by **Essential C++** to prepare the engineer for the practical use of SystemC for transaction-level modeling. The course describes the core SystemC v2.2 class library and its application for modeling systems, communication, hardware and software at the transaction-level, and refinement towards hardware-software implementation.

The details of the OSCI TLM-2.0 standard are covered by a separate follow-on course, **SystemC Modeling using TLM-2.0**

The workshops are based around carefully designed exercises to reinforce and challenge the extent of learning, and comprise approximately 50% of class time. Delegates can use the tools and platform of their choice on all exercises and workshops.

Doulos has a world-wide lead in independent SystemC know-how having been active in SystemC-based methods since 2000. We have delivered SystemC training and support to engineers in more than 170 companies world-wide – including direct involvement with methodology and tool developers in such companies as ARM, Cadence, CoWare, Mentor Graphics and Synopsys.

## Who should attend?

- Hardware design engineers who wish to become skilled in the practical use of SystemC for modeling digital hardware
- System engineers and architects who wish to become skilled in the practical use of SystemC for system level modeling
- Software engineers who already have good knowledge of C/C++, but who wish to acquire some practical experience in the use of the SystemC class libraries

## What will you learn?

- The C++ language features necessary to master SystemC
- Object-oriented programming techniques as used by the SystemC class libraries
- The SystemC core language, data types and channels
- How to make best use of the SystemC simulator to debug and validate your models
- How to move up from RTL modeling to transaction-level modeling

For further information contact your local Doulos [Sales Office](#).



# Comprehensive SystemC

## Standard Level - 5 days

- How to refine SystemC models between levels of abstraction
- An overview of the OSCI TLM-1.0 and TLM-2.0 standards
- An overview of hardware synthesis using SystemC (optional)
- An overview of the SystemC Verification Library SCV (optional)

## Pre-requisites

**Essential C++ for SystemC** - Delegates need basic knowledge of the C programming language, in particular familiarity with C functions, variables, data types, operators, and statements. This module is suitable for people with no previous knowledge of C++, as a refresher for those with limited knowledge of C++, or for hardware engineers who are familiar with VHDL or Verilog®.

**Fundamentals of SystemC** - A working knowledge of C++ and of object-oriented programming concepts is essential and basic knowledge of hardware design is recommended. Prior attendance of the Doulos 'Essential C++' class (or equivalent) is required. Delegates with C++ experience should check their knowledge against the '**SystemC C++ Pre-requisites**' (available from Doulos) before attending. The class is suitable for electronic hardware, software or systems engineers, but in order to gain maximum benefit from this class delegates should be active users of either a high-level software programming language (ideally C++) or a hardware description language (VHDL or Verilog®)

Please contact Doulos direct to discuss and assess your specific experience against the pre-requisites.

## Training materials

Doulos class materials are renowned for being the most comprehensive and user friendly available. Their style, content and coverage is unique in the EDA training world, and has made them sought after resources in their own right. Fees include

- Fully indexed class notes creating a complete reference manual
- Workbook full of practical examples and solutions to help you apply your knowledge
- Doulos SystemC Golden Reference Guide for language, syntax, semantics and tips.

## Structure and content

### Essential C++ for SystemC (2 days)

#### Day 1

Learn about the differences between C and C++

#### From C to C++

The features added to C by C++ and the ANSI C-1999 standard • `const` • `bool` • Header files • Namespaces • The global and standard namespaces • Stream I/O

#### Functional and Pointers

Learn how functions and dynamic memory allocation have changed in C++ • Pass-by-reference • Function prototypes • Default arguments • Function overloading • Operator overloading • Static, automatic and dynamic storage • `new` • `delete`

For further information contact your local Doulos [Sales Office](#).



# Comprehensive SystemC

## Standard Level - 5 days

### The C++ Standard Library

Learn to make the most of the built-in standard classes • Container classes • Examples of using the standard vector class • Examples of using the standard string and stringstream classes

### Classes and Objects

Learn the principles of object-based design • Information hiding • Abstract data types • Classes and objects • Public and private class members • Member functions • Scope resolution

### Day 2

Master the subtleties of object-oriented programming in C++

### Class Members

Master the C++ mechanisms associated with classes • Constructors • Destructors • Copy constructors • Pointers versus objects • Friends • `this` • Overloading operators as members • Static members • Constant objects and members

### Inheritance

Learn to exploit the power of object-oriented programming • Class relationships • Initializing sub-objects • The default constructor • Derived classes • Inheritance • Protected members • Up- and down-casting • Order of initialization

### Virtual Functions

Delve deeper into object-oriented programming techniques • Overriding methods • Virtual functions • Polymorphism • Run-time type identification • Abstract base classes • Multiple inheritance

### Further C++ Features

Advanced C++ features used in the SystemC class libraries • Function templates • Class templates • Implicit conversions • User-defined conversions • Exceptions

## Fundamentals of SystemC (3 days)

### Day 3

Become proficient in using the features of SystemC

### Introduction to SystemC

Learn the background to SystemC and how SystemC fits into the system-level design flow • The architecture of the SystemC release • The benefits and risks of adopting SystemC • The objectives of transaction-level modeling

### Getting Started

Learn how SystemC source code is structured and how to organise files • SystemC header files and namespaces • Compiling and executing a SystemC model

### Modules and Hierarchy

How to describe the structural connections between modules • Modules • Ports • Processes • Signals • Methods • Primitive channels • Module instantiation • Port binding

# Comprehensive SystemC

## Standard Level - 5 days

### Processes and Time

Describing concurrency and the passage of time • SC\_METHOD • SC\_THREAD • Event finders • Static and dynamic sensitivity • Time • Events • Clocks • Dynamic processes

### The Scheduler

Gain an insight into how SystemC manages the scheduling of processes and events • Starting and stopping simulation • Elaboration and simulation callbacks • The phases of simulation • Event notification • Event queues • `wait` and `next_trigger`

### Day 4

Learn to apply SystemC to modeling data, communication and busses.

### Debugging and Tracing

Learn about the facilities provided by SystemC to ease debugging and diagnostics • Debugging techniques • The standard reporting mechanism • Error handling • Writing trace (`vcd`) files • Tracing buried signals and local variables • Using waveform display tools

### SystemC Data Types

Data types for bit-accurate and hardware modeling • Signed and unsigned integers • Limited and finite precision integers • Assignment and truncation • Type conversion • Bit and part selects • Concatenation • Bit and logic vectors • Hexadecimal numbers • Avoiding common pitfalls • Bus resolution • Fixed point types

### Interfaces and Channels

Learn how channels are used to abstract communication and create fast simulation models • Hierarchical, primitive and minimal channels • Interface method calls • SystemC interfaces • Port-less channel access • The SystemC object hierarchy • The class `sc_port` • Registering ports • How to make the most of ports, channels and interfaces

### Day 5

Exploration of the application of Transaction-Level Modeling

### Bus Modeling

Learn the techniques required to write and use bus models in SystemC • Master and slave interfaces • The execution context of interface method calls • Blocking and non-blocking methods • Using events and dynamic sensitivity within channels • Multi-ports • Port binding policies • `sc_export`

### Refinement

An example of refinement from a C algorithm through untimed and timed SystemC models down to a mixed hardware-software implementation • SystemC wrappers • Timing annotation • Using `sc_buffer` • Structural refinement, communication refinement, and data refinement

### Adapters

Channel refinement using adapters • Events versus event finders • Instantiating and binding adapters

Continued...

# Comprehensive SystemC

## Standard Level - 5 days

### Transaction-Level Modeling

The OSCI Transaction-Level Modeling Standards • TLM-1.0 method calls • Unidirectional interfaces • TLM-2.0 requirements • Abstraction levels • Use cases • Coding styles • The architecture of TLM-2.0 • The interoperability layer • Utilities • Initiators, targets, and interconnect • Generic payload • Extensions

### Supplementary Subjects

#### Fixed Point Types

Fixed point word length and integer word length • Quantization modes • Overflow modes • Fixed point context • The type cast switch • Utility methods

#### Overview of SystemC Synthesis

RTL versus behavioural synthesis technology • The work of the OSCI synthesis working group • Synthesizable data types • Clocked threads and resets • Restrictions

#### Overview of the SystemC Verification Library

Introduction to and aims of SCV • Constrained random verification methodology • Extended data types to support introspection • Randomization • Transaction Recording

### Related classes

- Expert SystemC Modeling
- Expert SystemC Verification
- Modular SystemC (**in-house delivery only**)
- Comprehensive C++ (5-days)
- Essential Digital Design Techniques

### Project services

Doulos Project Services enable clients to access our world-leading technical know-how and apply it directly to projects. **Expert-on-call**, **Expert-design** and **Expert-support** options can be flexibly packaged and delivered to provide valuable expertise and additional resource just when it is needed.

### How to book a class

To make a provisional booking, or to obtain pricing information, please contact your local Doulos sales team. You will find contact details on our [website](#).

For further information contact your local Doulos [Sales Office](#).

